REGISTER ACCESS PROTOCOL FOR MULTI PROCESSOR SYSTEMS

BACKGROUND

[0001] Multi-processor systems, e.g. MODEMS, networking peripherals, are becoming commonplace as processor or digital signal processing (DSP) based I/Os are added to systems. The processors may be included within a single chip or located in separate interconnected integrated circuits (ICs). The multi-processor system can communicate via registers, as shown in Figures 1 and 2. Care must be taken to prevent the processors from inadvertently corrupting the registers, e.g. concurrent attempts to read-modify-write within the same register. Unless an explicit contention management technique is employed, the registers may become corrupted.

[0002] Prior art contention management schemes rely on software, semaphores, or fixed priority hardware arbitration within the registers. The schemes are often error prone, inflexible, and difficult to verify.

SUMMARY

[0003] The present invention provides shared registers in a multi-processor system with a contention management protocol when a register is simultaneously accessed by more than one processor.

[0004] Each register includes access protocol and data. The access protocol includes an access type for each processors and arbitration priority. The access type being selected from a group that includes READ, READ/CLEAR, READ/SET, and READ/WRITE.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] Figure 1 illustrates a multi-processor system of the prior art.

[0006] Figure 2 illustrates a multi-processor system of the prior art.

[0007] Figure 3 illustrates an illustrative example of the configuration register of the present invention.

DETAILED DESCRIPTION

[0008] The present invention is a contention management protocol that is hardware configurable at run-time for flexibility and robustness. For each element, e.g. bit, register, or bank of registers, the programmable contention management specifies what level of write access each processor has to the specified element.

[0009] The access types are defined: READ, READ/CLEAR, READ/SET, and READ/WRITE. For READ, the processor may only read the element. Attempts to write are ignored or optionally generate an error. For READ/CLEAR, the processor may read or clear bits within the element. For READ/SET, the processor may read or set bits within the element. For READ/WRITE, the processor may read, set, or clear bits within the element.

[0010] The access types permit simple handshaking protocols to be implemented between processors. For example, a simple interlocked handshake is implemented when one processor is only allowed to set a particular element and a second processor is only allowed to clear it.

[0011] In addition to the access type, an optional arbitration priority is also specified. This defines what happens when there is a conflict between processor access types. To illustrate, if one processor tries to READ/CLEAR a particular element while another processor tries to READ/SET the same element, the arbitration priority defines the outcome. For systems implemented without arbitration priority, the access types would be configured as mutually exclusive.

[0012] In one illustrative embodiment for a two-processor system, 5 configuration bits may be used for each element as shown in Table 1.

Table 1

accessTypeCpu1	READ=00, READ/CLEAR= 01, READ/SET=10, READ/WRITE = 11
accessTypeCpu2	READ=00, READ/CLEAR= 01, READ/SET=10, READ/WRITE = 11
ArbitrationPriority	Processor1=0, Processor2=1
Configuration[4:0]	{ArbitrationPriority, AccessTypeCpu1, AccessTypeCpu2)

[0013] Hence, a value of "10110" would indicate that processor1 has READ/CLEAR access and Processor2 has READ/SET access. In the event of a conflict, Processor2 has priority of Processor1. This might be used for an interrupt from Processor2 to processor1. Processor2 sets the interrupt bit and it remains set until Processsor1 acknowledges clearing it.

[0014] Figure 3 illustrates an illustrative example of the data within a shared system or configuration register. In this embodiment, the most significant five bits are used for access control. The remaining bits in the register store the data.

[0015] The invention allows configuration of a wide range of interface protocols to permit hardware to be developed before a protocol is known and to allow hardware to be reconfigured to support a different protocol. The most generic implementation would provide the configuration bits for each bit in a register. However, a lower cost implementation is possible by protecting collections of bits or registers. The configuration bits can be controlled by a single processor or jointly. Alternatively, the access protocol may be encoded and selected as a build-time option in the hardware design source code or encoded and provided as input signals to the hardware design.

[0015] One with skill in the art may extend the inventive concept. For an multi processor system having N processors, where N is an integer, $N \geq 2$, each programmable configuration register consists of 2N bits, where each of the configurable access types are encoded into 2 bits. When the optional arbitration priority is included, an additional $N*\text{ceiling}(\log_2 N)$ bits may be used, each set of $\text{ceiling}(\log_2 N)$ bits provides for relative arbitration priority of each processor. The ceiling function is defined as follows: for any given real number x, ceiling(x) is the smallest integer no less than x. For example, in a 8-processor system, the relative priority of each processor could be encoded in 3 bits. If each processor is encoded with a unique arbitration priority number, the logic can determine which processor has write priority. In this example case, 5 bits would be required per processor, or a total of 40 bits for each shared register element.